UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/848,869	05/18/2004	Richard Yu Gu	ORA018 US	6186

56135 7590 12/22/2008

Silicon Valley Patent Group LLP
18805 Cox Avenue
SUITE 220
Saratoga, CA 95070

EXAMINER	
VU, BAI D	

ART UNIT	PAPER NUMBER
2165	

MAIL DATE	DELIVERY MODE
12/22/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/848,869	GU ET AL.	
	Examiner	Art Unit	
	Bai D. Vu	2165	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *14 July 2008 and 21 September 2008*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-11 and 13-20* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-11 and 13-20* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

Response to Amendment

1. Applicant has amended claims 1-11 and 13-20 in the amendment filed on 09/21/2008.

Claims 1-11 and 13-20 are pending in this office action.

Response to Arguments

2. Applicant's arguments filed on 07/14/2008 with respect to claims 1-11 and 13-20 have been considered but are moot in view of the new ground(s) of rejection.

Regarding to the objection of specification:

After review the claimed amendments supported by the amended specification mailed on 11/23/2007, the objection to the specification under 37 CFR 1.75(d)(1) (see MPEP § 608.01(o)) of failing to provide antecedent basis for the claim limitation "writing a plurality of logs, at least one log" is withdrawn. However, examiner fails to find any support of the drawings corresponding to the amended specification; therefore, the examiner hereby withdraws the last office action mailed on 03/12/2008 and replaces with rejection of claim 9 under 35 U.S.C. § 112, first paragraph in this office action as addressed below.

Regarding to the applicant's argument:

In response to applicant's argument, examiner will fully reconsider the claimed amendments filed on 09/21/2008 after the clarification of the 35 U.S.C. § 112, first paragraph rejection is considered in the reply to this office action.

Specification

3. The amended specification filed on 11/23/2007 is objected to under 35 U.S.C. 132(a) because: the limitation "writing a plurality of logs, at least one log for each row identified in the group of row-identifier and value pairs" claim 9 is not provided in the original specification mailed on 05/18/2004; also, the filed specification on 11/23/2007 introduces new matter into the disclosure. 35 U.S.C. 132(a) states that no amendment shall introduce new matter into the disclosure of the invention. The added material which is not supported by the original disclosure is as follow: *In some embodiments, the method further comprises writing a plurality of logs, at least one log for each row identified in the group of row-identifier and value pairs.*

Applicant is required to cancel the new matter in the reply to this office action.

Claim Rejections - 35 USC § 112

4. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

5. **Claim 9** is rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention.

The claim limitation "*writing a plurality of logs, at least one log for each row identified in the group of row-identifier and value pairs and performing a write operation from said cache to said storage device when space is needed in said cache*" in claim 9 lines 2-4, contains subject matter which was not described in the original specification mailed on 05/18/2004 in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention.

Claim Rejections - 35 USC § 103

6. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

7. **Claims 1-5, 7, 8, 11, 13, 14, 18 and 19** are rejected under 35 U.S.C. 103(a) as being unpatentable over Agarwal et al. (US Pat. No. 7,080,081 B2) in view of Machado et al. (US Pat. No. 5,517,631).

As per **claim 1**, Agarwal et al. discloses "a method implemented in a computer, the method comprising:"

"generating an array update operation based on a query to update a relational database; wherein said array update operation specifies a plurality of row-identifier and value pairs to update multiple rows in a table of said relational database;"

"repeatedly finding, and storing in a structure, a block-identifier of a block that contains a row of data identified by a row-identifier in at least a group of row-identifier and value pairs in said plurality, by use of a database index prior to retrieval of the block" as cited herein *FIGS. 3-5 and 8 interpreted as structure*; *processing a query further includes using information from either the individual block indexes or the composite index to obtain a list of block identifier, and scanning blocks of the table for records (col. 2 lines 42-45)*; *processing a query includes the steps of scanning the entire table for records and using a record-based index to find records (col. 2 lines 46-48)* wherein the records interpreted as a group of row-identifier and value pairs; and *typically, the client/user issues SQL commands that are processed by the RDBMS 103 and results are returned. During operation, the query compiler 201 parses the input SQL commands and uses the code generator 202 to generate an execution plan. The parsed SQL commands are typically transformed into an internal representation and are then optimized. Optimization involves looking at several alternative strategies for obtaining the correct result, and choosing the most efficient strategy. The execution engine 203 interprets and executes the plan and produces the desired results. The execution engine 203 submits requests to the data manager 207 to obtain information from tables.*

This is done in the manner that was determined by the query compiler 201 (or separate optimizer), using available indexes, scanning tables, etc. The execution engine 203 uses the access methods engine 204 to efficiently access the underlying database tables that are stored in the access methods engine 204 (or externally thereto). The relevant data items are then retrieved and stored in the buffer manager 205 for reusability of the data (col. 4 line 664 to col. 5 line 16) explains obtaining list of block identifiers prior to retrieving the blocks.

"wherein said structure is located in main memory of said computer;"

"wherein each value comprises data to be updated in said row identified by said row-identifier;" as cited herein *in the exemplary multidimensional table depicted in FIG. 4, to find the slice containing all records with `ON` for the Province dimension, we would look up this key value in the Province dimension block index, and find a key such as the following: <ON: 9, 16, 18, 19, 22, 24, 25, 30, 36, 39, 41, 42> where the key is in the form of a <key value: BID(s)> pair (col. 6 lines 43-50).*

performing a single access operation without context witching, to access retrieve from a storage device and store in a cache, a number of blocks of data of said table, said blocks being identified by block-identifiers in the structure; (as cited herein *the execution engine 203 uses the access methods engine 204 to efficiently access the underlying database tables that are stored in the access methods engine 204 (or externally thereto). The relevant data items are then retrieved and stored in the buffer manager 205 for reusability of the data. Typically, relational database management*

systems provide sequential table scan access as well as index-based access to tables (col. 5 lines 11-18)).

"wherein several of said blocks are non-contiguous in said storage device; and"

"repeatedly updating, in blocks in the cache, each row identified in the group of row-identifier and value pairs, using a corresponding value in the row-identifier and value pairs" as cited herein *Insert, Delete, Purge, and Update functions (col. 11 line 4 to col. 12 line 55)*.

Agarwal et al. does not explicitly disclose "performing a single access operation without context witching". However, Machado et al. discloses as *the programmable data sequencer comprises a writeable control store including a random access memory area directly addressable by the programmed digital microcontroller for writing sequences of control patterns, there being most preferably a single sequence written for controlling states of the programmable data sequencer during both data read and data write operation to and from the disk surface and a buffer memory (5 lines 33-40)*.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to apply Machado et al. teaching of automatic data block sequencing with dual function opcodes for reading and writing data blocks in real time into Agarwal et al. system in order to provide a data sequencer for a disk drive employing zoned data recording having data fields split into segments by intervening embedded servo sectors and wherein the data sequencer provides for automatic sequencing of data blocks during writing data to, and reading data from, the split data fields (Machado et al., col. 3 line 63 to col. 4 line 1).

As per **claim 2**, Agarwal et al. discloses "the method of claim 1 further comprising:"

"sorting the block-identifiers, prior to retrieval of the blocks" as cited herein *the relevant data items are then retrieved and stored in the buffer manager 205 for reusability of the data. Typically, relational database management systems provide sequential table scan access as well as index-based access to tables (col. 5 lines 14-18)*; and *while blocks are numbered sequentially starting from block 1 in the exemplary table shown herein, it should be appreciated that the blocks could be identified in numerous other ways (col. 6 lines 10-13).*

Agarwal et al. does not explicitly disclose "performing the single access operation". However, Machado et al. discloses as *the programmable data sequencer comprises a writeable control store including a random access memory area directly addressable by the programmed digital microcontroller for writing sequences of control patterns, there being most preferably a single sequence written for controlling states of the programmable data sequencer during both data read and data write operation to and from the disk surface and a buffer memory (col. 5 lines 33-40).*

It would have been obvious to one of ordinary skill in the art at the time the invention was made to apply Machado et al. teaching of automatic data block sequencing with dual function opcodes for reading and writing data blocks in real time into Agarwal et al. system in order to provide a data sequencer for a disk drive employing zoned data recording having data fields split into segments by intervening

embedded servo sectors and wherein the data sequencer provides for automatic sequencing of data blocks during writing data to, and reading data from, the split data fields (Machado et al., col. 3 line 63 to col. 4 line 1).

As per **claim 3**, Agarwal et al. discloses "the method of claim 2 wherein: the sorting is performed subsequent to storage of the block-identifiers in the structure" as cited herein *FIGS. 3-5 and 8 interpreted as structure*; *processing a query further includes using information from either the individual block indexes or the composite index to obtain a list of block identifier, and scanning blocks of the table for records (col. 2 lines 42-45)*; *processing a query includes the steps of scanning the entire table for records and using a record-based index to find records (col. 2 lines 46-48)*; and *while blocks are numbered sequentially starting from block 1 in the exemplary table shown herein, it should be appreciated that the blocks could be identified in numerous other ways (col. 6 lines 10-13)*.

As per **claim 4**, Agarwal et al. discloses "a method of claim 1 further comprising: subsequent to said finding and prior to said storing, checking if the block identifier has a duplicate already stored in the structure and if so then not storing the block identifier in the structure" as cited herein *mentioned, a block-based index ORing operation may also be performed using block indexes. For example, if the query includes the condition Province=`ON` or Province=`BC`, then the province block index can be scanned for each category and an aggregated list of blocks can be obtained by an ORing operation.*

The ORing operation can eliminate duplicate BIDs which are possible for conditions such as Province=`AB` or Color=`Red` (col. 8 line 62 to col. 9 line 2).

As per **claim 5**, Agarwal et al. discloses "the method of claim 1 further comprising, prior to updating: repeating said finding of block-identifiers for all row-identifiers in the group of row-identifier and value pairs" as cited herein *processing a query further includes using information from either the individual block indexes or the composite index to obtain a list of block identifier, and scanning blocks of the table for records (col. 2 lines 42-45).*

As per **claim 7**, Agarwal et al. discloses "the method of claim 1 wherein: the database index is a B-tree index" as cited herein *typically, relational database management systems provide sequential table scan access as well as index-based access to tables. The B-Tree index is the most preferred index technique in RDBMS systems. Optionally, some RDBMS systems allow that the underlying data be clustered and/or partitioned using one or more columns (or index) (col. 5 lines 16-22).*

As per **claim 8**, Agarwal et al. discloses "the method of claim 1 wherein: said structure comprises an array; and the array has a number of entries identical to the number of blocks that can be held in the cache" as cited herein *see FIG. 8 and a block index is created for each dimension of the table. According to another aspect of the invention, the blocks store information in a contiguous storage space (col. 2 lines 4-7).*

As per **claim 11**, Agarwal et al. discloses "the method of claim 1 wherein a plurality of file offsets are provided to the single access operation, one file offset for each block in the group;" as cited herein *each block may have a header, located in a first slot of the block's first page which stores a structure containing, among other possible things, a copy of the block status so that the block map can be re-created if necessary in case of deletion or corruption of the map, and a bit map covering the pages of the block, indicating which pages are empty (e.g., 0=empty, 1=nonempty, even if it contains only overflow or pointer records). Each block may also have a free space control record (FSCR) associated with it that could contain page offsets and approximations of free space per page. These FSCR's may be located on the first page of a block, and stored as the second record on this page, for example (col. 9 lines 40-51).*

"wherein each file offset is an offset in a file from where reading of data is to begin".

Agarwal et al. does not explicitly disclose "the single access operation". However, Machado et al. discloses as *the programmable data sequencer comprises a writeable control store including a random access memory area directly addressable by the programmed digital microcontroller for writing sequences of control patterns, there being most preferably a single sequence written for controlling states of the programmable data sequencer during both data read and data write operation to and from the disk surface and a buffer memory (5 lines 33-40).*

It would have been obvious to one of ordinary skill in the art at the time the invention was made to apply Machado et al. teaching of automatic data block sequencing with dual function opcodes for reading and writing data blocks in real time into Agarwal et al. system in order to provide a data sequencer for a disk drive employing zoned data recording having data fields split into segments by intervening embedded servo sectors and wherein the data sequencer provides for automatic sequencing of data blocks during writing data to, and reading data from, the split data fields (Machado et al., col. 3 line 63 to col. 4 line 1).

As per **claim 13**, Agarwal et al. discloses "a non-volatile media in which are stored instructions to perform a method comprising:"

"generating an array update operation based on a query to update a relational database; wherein said array update operation specifies a plurality of row-identifier and value pairs to update multiple rows in a table of said relational database;"

"repeatedly finding, and storing in a structure, a block-identifier of a block that contains a row identified by a row-identifier in at least a group of row-identifier and value pairs in said plurality, by use of a database index of a relational database" as cited herein *FIGS. 3-5 and 8 interpreted as structure*; *processing a query further includes using information from either the individual block indexes or the composite index to obtain a list of block identifier, and scanning blocks of the table for records (col. 2 lines 42-45)*; and *processing a query includes the steps of scanning the entire table for*

records and using a record-based index to find records (col. 2 lines 46-48) wherein the records interpreted as a group of row-identifier and value pairs.

performing a vector read operation without context witching during said performing, to retrieve from a storage device and store in a cache, a number of blocks, said blocks being identified by block-identifiers in the structure; and (as cited herein *the execution engine 203 uses the access methods engine 204 to efficiently access the underlying database tables that are stored in the access methods engine 204 (or externally thereto). The relevant data items are then retrieved and stored in the buffer manager 205 for reusability of the data. Typically, relational database management systems provide sequential table scan access as well as index-based access to tables (col. 5 lines 11-18)).*

"repeatedly updating, in blocks in the buffer cache, each row identified in the group of row-identifier and value pairs, using a corresponding value in the row-identifier and value pairs;" as cited herein *Insert, Delete, Purge, and Update functions (col. 11 line 4 to col. 12 line 55).*

"wherein several of said blocks are non-contiguous in said storage device".

Agarwal et al. does not explicitly disclose "performing a vector read operation without context witching". However, Machado et al. discloses as *the programmable data sequencer comprises a writeable control store including a random access memory area directly addressable by the programmed digital microcontroller for writing sequences of control patterns, there being most preferably a single sequence written for controlling*

states of the programmable data sequencer during both data read and data write operation to and from the disk surface and a buffer memory (5 lines 33-40).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to apply Machado et al. teaching of automatic data block sequencing with dual function opcodes for reading and writing data blocks in real time into Agarwal et al. system in order to provide a data sequencer for a disk drive employing zoned data recording having data fields split into segments by intervening embedded servo sectors and wherein the data sequencer provides for automatic sequencing of data blocks during writing data to, and reading data from, the split data fields (Machado et al., col. 3 line 63 to col. 4 line 1).

As per **claim 14**, Agarwal et al. discloses "a non-volatile media of claim 13 being further encoded with said structure storing the block-identifiers" as cited herein *the load utility can access a data set formatted in a particular manner, and use the information in the data set to create rows in a particular table (col. 10 lines 4-6).*

As per **claim 18**, Agarwal et al. discloses "a non-volatile media of claim 13 being comprised in at least one of a disk, a chip and cartridge" as cited herein *it is to be appreciated that the term "processor" as used herein is intended to include any processing device, such as, for example, one that includes a CPU (central processing unit). The term "memory" as used herein is intended to include memory associated with a processor or CPU, such as, for example, RAM, ROM, a fixed memory device (e.g.,*

hard drive), a removable memory device (e.g., diskette), etc. In addition, the term "input/output devices" or "I/O devices" as used herein is intended to include, for example, one or more input devices, e.g., a keyboard, for making queries and/or inputting data to the processing unit, and/or one or more output devices, e.g., CRT display and/or printer, for presenting query results and/or other results associated with the processing unit. It is also to be understood that various elements associated with a processor may be shared by other processors. Accordingly, software components including instructions or code for performing the methodologies of the invention, as described herein, may be stored in one or more of the associated memory devices (e.g., ROM, fixed or removable memory) and, when ready to be utilized, loaded in part or in whole (e.g., into RAM) and executed by a CPU (col. 4 lines 22-42).

As per **claim 19**, Agarwal et al. discloses "the method of claim 2 wherein: the blocks are sorted during said sorting based on adjacency such that during performance of said single access operation, block-identifiers of blocks physically adjacent to one another at a periphery of a disk in the storage device are presented at one time to the storage device and identifiers of blocks that are physically adjacent to one another and located closer to a center of the disk are presented at another time" as cited herein *the relevant data items are then retrieved and stored in the buffer manager 205 for reusability of the data. Typically, relational database management systems provide sequential table scan access as well as index-based access to tables (col. 5 lines 14-18)*; and *while blocks are numbered sequentially starting from block 1 in the exemplary*

table shown herein, it should be appreciated that the blocks could be identified in numerous other ways (col. 6 lines 10-13).

8. **Claims 9 and 10** are rejected under 35 U.S.C. 103(a) as being unpatentable over Agarwal et al. in view of Machado et al., and further in view of Debrunner (US Pat. No. 6,321,234 B1).

As per **claim 9**, Agarwal et al. and Machado et al. do not explicitly disclose "the method of claim 1 further comprising: writing a plurality of logs, at least one log for each row identified in the group of row-identifier and value pairs and performing a write operation from said cache to said storage device when space is needed in said cache". However, Debrunner discloses as *transactions are processed by the system, Logging System 270 is used to log changes which occur to the system. In the commercial embodiment of Sybase SQL Server System 11, this is done by posting log records to a transaction log. Every transactional update to an object, including inserts, updates, and deletes, causes a log record to be written to the transaction log or simply "log". In SQL Server, the transaction log is referred to as "syslogs." Each particular log record characterizes the change which has occurred to the database during processing of a transaction. This information can be used, for instance, in error recovery, to restore the database to a pre-existing, consistent state (col. 7 lines 18-29)*; *when a transaction commits or the memory fills with log records, the PLC associated with the transaction is flushed to the log (col. 3 lines 22-24)*; and *the log itself exists in two versions: an in-*

memory version and a disk version. The in-memory version of the log exists as page chain in system memory. Here, data pages storing log records are linked together in memory to form a chain of pages. The in-memory log is flushed, at appropriate times, to disk for creating the disk version of the log. In typical operation, when a transaction "commits," the log records must first be written to disk before the database system proceeds with actually committing the transaction (col. 7 lines 50-58).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to apply Debrunner teaching of process of logging transactions into Agarwal et al. and Machado et al. systems in order to preserve database throughput by reducing the contention which occurs for logging resources, even when such a system is scaled up with multiple database engines (Debrunner, col. 3 lines 7-10).

As per **claim 10**, Agarwal et al. and Machado et al. do not explicitly disclose "the method of claim 9 further comprising, during said write operation: unpinning each block after updating all rows in said each block; and flushing an unpinned block to disk only when another block needs space in the cache occupied by the unpinned block". However, Debrunner discloses as *recall that the "log pin" represents where in the log page chain log records must be flushed to disk before the corresponding data page can be flushed. Besides this mechanism, the present invention introduces a second mechanism. Because log records can reside in a (user) private log cache (i.e., per "user" connection), a mechanism is needed to ensure that such records are flushed (to the log page chain). If a change which affects a data page is represented in a private*

log cache, the PLC pin for that page will be instantiated and will point to the corresponding PLC_DATA data structure (through the xdes data structure). In this manner, when such a data page is written, the system first unpins the PLC pin, thus causing the corresponding PLC_DATA data structure to be flushed to the log page chain. Now, the data page will get a pin to the log, for indicating where in the log records must be flushed in order to write the data page. The addition of the second pointer to the data page buffer provides a two-phase unpinning mechanism. Unpinning a page now requires that the corresponding private log cache (i.e., the one being pointed to by the pointer stored in the page) be first flushed to the log page chain. Secondly, the in-memory log itself is flushed to disk (up to the point in the log required for writing that data page) (col. 13 lines 53-67 and col. 14 lines 33-40).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to apply Debrunner teaching of process of logging transactions into Agarwal et al. and Machado et al. systems in order to preserve database throughput by reducing the contention which occurs for logging resources, even when such a system is scaled up with multiple database engines (Debrunner, col. 3 lines 7-10).

9. **Claims 15-17 and 20** are rejected under 35 U.S.C. 103(a) as being unpatentable over Agarwal et al. in view of Machado et al., and further in view of Duvillier et al. (US Pub. No. 2002/0103815 A1).

As per **claim 15**, Agarwal et al. discloses "a computer comprising a processor and a memory coupled to the processor, the memory being encoded with instructions to:"

"automatically generate an array update operation based on a query to update a relational database;"

"automatically use a database index to look up a block identifier of a block that contains a row identified by an identifier in a plurality of identifier and value pairs to be used to perform said array update operation on a table in said relational database" as cited herein *processing a query further includes using information from either the individual block indexes or the composite index to obtain a list of block identifier, and scanning blocks of the table for records (col. 2 lines 42-45).*

"automatically store the block identifier in a structure in memory" as cited herein *FIGS 3-5 and 8*;

"automatically repeat instructions to said automatically use and said automatically store, for all identifiers in at least a group of identifier and value pairs in said plurality;" as cited herein *processing a query includes the steps of scanning the entire table for records and using a record-based index to find records (col. 2 lines 46-48)* wherein the records interpreted as a group of row-identifier and value pairs.

automatically perform a vector read, to retrieve from a disk and store in a cache, each block in a group of blocks identified by block-identifiers stored in said structure, wherein the group of blocks are all stored in the cache during execution of a single function call; (as cited herein *the execution engine 203 uses the access methods engine*

204 to efficiently access the underlying database tables that are stored in the access methods engine 204 (or externally thereto). The relevant data items are then retrieved and stored in the buffer manager 205 for reusability of the data. Typically, relational database management systems provide sequential table scan access as well as index-based access to tables (col. 5 lines 11-18)).

automatically modify a row in a block stored in the cache, using a value in the plurality of identifier and value pairs; automatically repeat instructions to said automatically modify with each row identified in the group of identifier and value pairs (as cited herein *since each block contains potentially many pages of records, these block indexes are much smaller than RID indexes and need only be updated as new blocks are required and so added to a cell, or existing blocks are emptied so as to be removed from a cell (col. 6 lines 33-38)*).

Agarwal et al. does not explicitly disclose "data stored in the cache during execution of a single function call and modifying a row stored in the cache".

However, Machado et al. discloses as *the programmable data sequencer comprises a writeable control store including a random access memory area directly addressable by the programmed digital microcontroller for writing sequences of control patterns, there being most preferably a single sequence written for controlling states of the programmable data sequencer during both data read and data write operation to and from the disk surface and a buffer memory (5 lines 33-40)*. Duvillier et al. discloses as *each time a particular object in the database is updated or modified, a new version of that object is created. When the new object version is created, a copy of the new object*

version is stored in a disk page buffer in the data server cache (¶ 0140 lines 1-6) wherein the object interpreted as a row.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to apply Machado et al. teaching of automatic data block sequencing with dual function opcodes for reading and writing data blocks in real time into Agarwal et al. system in order to provide a data sequencer for a disk drive employing zoned data recording having data fields split into segments by intervening embedded servo sectors and wherein the data sequencer provides for automatic sequencing of data blocks during writing data to, and reading data from, the split data fields (Machado et al., col. 3 line 63 to col. 4 line 1). Furthermore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to apply Duvillier et al. teaching of a technique for improving performance of information storage and retrieval systems into Agarwal et al. and Machado et al. systems in order to utilize logical addresses for mapping object locations and physical addresses of objects stored within the data structures of the system (Duvillier et al., ¶ 0053 lines 8-10).

As per **claim 16**, Agarwal et al. discloses "an apparatus comprising a database, the apparatus comprising:"

"means for generating an array update operation based on a query to update the database; wherein said array update operation specifies a plurality of row-identifier and value pairs to update multiple rows in a table of the database;"

"means for using a database index to look up a block identifier of a block that contains the row identified by an identifier in the plurality of identifier and value pairs;" as cited herein *processing a query further includes using information from either the individual block indexes or the composite index to obtain a list of block identifier, and scanning blocks of the table for records (col. 2 lines 42-45).*

"means for storing the block-identifier in a structure in memory;" as cited herein *FIGS 3-5 and 8*;

"means for repeating (using the database index to look up and storing the block-identifier) for all identifiers in at least a group of identifier and value pairs;" as cited herein *processing a query includes the steps of scanning the entire table for records and using a record-based index to find records (col. 2 lines 46-48)* wherein the records interpreted as a group of row-identifier and value pairs.

means for performing a vector read, to store in a cache, each block in a group of blocks identified by block-identifiers stored in said structure, wherein the group of blocks are all stored in the cache during execution of a single function call (as cited herein *the execution engine 203 uses the access methods engine 204 to efficiently access the underlying database tables that are stored in the access methods engine 204 (or externally thereto). The relevant data items are then retrieved and stored in the buffer manager 205 for reusability of the data. Typically, relational database management systems provide sequential table scan access as well as index-based access to tables (col. 5 lines 11-18)*).

means for modifying a row in a block stored in the cache, using a value in the plurality of identifier and value pairs; and means for repeating said modifying with each row identified in the group of identifier and value pairs (as cited herein *since each block contains potentially many pages of records, these block indexes are much smaller than RID indexes and need only be updated as new blocks are required and so added to a cell, or existing blocks are emptied so as to be removed from a cell (col. 6 lines 33-38)*).

Agarwal et al. does not explicitly disclose "data stored in the cache during execution of a single function call and modifying a row stored in the cache".

However, Machado et al. discloses as *the programmable data sequencer comprises a writeable control store including a random access memory area directly addressable by the programmed digital microcontroller for writing sequences of control patterns, there being most preferably a single sequence written for controlling states of the programmable data sequencer during both data read and data write operation to and from the disk surface and a buffer memory (5 lines 33-40)*. Duvillier et al. discloses as *each time a particular object in the database is updated or modified, a new version of that object is created. When the new object version is created, a copy of the new object version is stored in a disk page buffer in the data server cache (¶ 0140 lines 1-6)* wherein the object interpreted as a row.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to apply Machado et al. teaching of automatic data block sequencing with dual function opcodes for reading and writing data blocks in real time into Agarwal et al. system in order to provide a data sequencer for a disk drive

employing zoned data recording having data fields split into segments by intervening embedded servo sectors and wherein the data sequencer provides for automatic sequencing of data blocks during writing data to, and reading data from, the split data fields (Machado et al., col. 3 line 63 to col. 4 line 1). Furthermore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to apply Duvillier et al. teaching of a technique for improving performance of information storage and retrieval systems into Agarwal et al. and Machado et al. systems in order to utilize logical addresses for mapping object locations and physical addresses of objects stored within the data structures of the system (Duvillier et al., ¶ 0053 lines 8-10).

As per **claim 17**, Agarwal et al. discloses "a method implemented in a computer, the method comprising:"

"generating an array update operation based on a query to update a database; wherein said array update operation specifies a plurality of row-identifier and value pairs to update multiple rows in a table of said database;"

"finding a block-identifier of a block that contains the row identified by a row-identifier in a row-identifier and value pair in said plurality, by use of a database index in said database;" as cited herein *processing a query further includes using information from either the individual block indexes or the composite index to obtain a list of block identifier, and scanning blocks of the table for records (col. 2 lines 42-45).*

"storing the block-identifier in a structure in memory;" as cited herein *FIGS 3-5 and 8*;

"repeating (finding the block-identifier and storing the block-identifier), for all row-identifiers in at least a group of row-identifier and value pairs in said plurality;" as cited herein *processing a query includes the steps of scanning the entire table for records and using a record-based index to find records (col. 2 lines 46-48)* wherein the records interpreted as a group of row-identifier and value pairs.

performing a vector read operation, to store in a buffer cache, each block in a group of blocks identified by block-identifiers stored in said structure, wherein the group of blocks are all stored in the cache during execution of a single function call (as cited herein *the execution engine 203 uses the access methods engine 204 to efficiently access the underlying database tables that are stored in the access methods engine 204 (or externally thereto). The relevant data items are then retrieved and stored in the buffer manager 205 for reusability of the data. Typically, relational database management systems provide sequential table scan access as well as index-based access to tables (col. 5 lines 11-18)*).

updating the row in the block in the cache, using the value in the row-identifier and value pair; and repeating said updating with each row identified in the group of row-identifier and value pairs (as cited herein *since each block contains potentially many pages of records, these block indexes are much smaller than RID indexes and need only be updated as new blocks are required and so added to a cell, or existing blocks are emptied so as to be removed from a cell (col. 6 lines 33-38)*).

Agarwal et al. does not explicitly disclose "data stored in the cache during execution of a single function call and modifying a row stored in the cache".

However, Machado et al. discloses as *the programmable data sequencer comprises a writeable control store including a random access memory area directly addressable by the programmed digital microcontroller for writing sequences of control patterns, there being most preferably a single sequence written for controlling states of the programmable data sequencer during both data read and data write operation to and from the disk surface and a buffer memory (5 lines 33-40)*. Duvillier et al. discloses as *each time a particular object in the database is updated or modified, a new version of that object is created. When the new object version is created, a copy of the new object version is stored in a disk page buffer in the data server cache (¶ 0140 lines 1-6)* wherein the object interpreted as a row.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to apply Machado et al. teaching of automatic data block sequencing with dual function opcodes for reading and writing data blocks in real time into Agarwal et al. system in order to provide a data sequencer for a disk drive employing zoned data recording having data fields split into segments by intervening embedded servo sectors and wherein the data sequencer provides for automatic sequencing of data blocks during writing data to, and reading data from, the split data fields (Machado et al., col. 3 line 63 to col. 4 line 1). Furthermore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to apply Duvillier et al. teaching of a technique for improving performance of information storage and retrieval systems into Agarwal et al. and Machado et al. systems in order to utilize

logical addresses for mapping object locations and physical addresses of objects stored within the data structures of the system (Duvillier et al., ¶ 0053 lines 8-10).

As per **claim 20**, Agarwal et al. discloses "the computer of claim 15 wherein: the blocks are sorted during said single function call based on adjacency such that block-identifiers of blocks physically adjacent to one another at a periphery of said disk are presented at one time to a disk drive comprising said disk and identifiers of blocks that are physically adjacent to one another and located closer to a center of said disk are presented at another time" as cited herein the relevant data items are then retrieved and stored in the buffer manager 205 for reusability of the data. Typically, relational database management systems provide sequential table scan access as well as index-based access to tables (col. 5 lines 14-18); and while blocks are numbered sequentially starting from block 1 in the exemplary table shown herein, it should be appreciated that the blocks could be identified in numerous other ways (col. 6 lines 10-13).

10. **Claim 6** is rejected under 35 U.S.C. 103(a) as being unpatentable over Agarwal et al. in view of Machado et al., and further in view of Vagnozzi (US Pat. No. 6,070,164).

As per **claim 6**, Agarwal et al. discloses "the method of claim 1 wherein: the database index is a hash index and the table is organized in a hash cluster; and" as cited herein *a block index is created for each dimension of the table. According to another aspect of the invention, the blocks store information in a contiguous storage*

space. According to another aspect of the invention, each block index comprises at least one key that is associated with a list of block identifiers (col. 2 lines 4-9).

Agarwal et al. and Machado et al. do not explicitly disclose "during said finding, a single directory is used to obtain the block identifier". However, Vagnozzi discloses as *referring to FIG. 1, there is shown a database 20 comprising a table of records 22. Each of the records has a fixed length and is assigned a unique record number, starting with zero. This allows the records to be stored sequentially in a single file with the location within the file of any particular record simply being determined by multiplying the record length (in bytes) by that record's assigned record number and using the resulting value as an offset from the beginning of the file. This arrangement provides very simple and fast allocation, de-allocation, and re-use of data record space within the file. Also, by storing all of the records within a single file, entire databases can be easily created or deleted. Creation simply requires creating a new, empty file. Deletion simply requires deleting an existing file. All user access to the database is by way of a database management program which allows the user to add, change, and delete data from the database (col. 4 lines 24-40)* explains storing all the records in a single file.

It would have been obvious to one of ordinary skill in the art at the time the invention was made to apply Vagnozzi teaching of single file directory into the Agarwal et al. and Machado et al. systems in order to provide very fast query response and fast update response time (Vagnozzi, col. 3 lines 1-2).

Conclusion

11. The examiner requests, in response to this Office Action, support is shown for language added to any original claims on amendment and any new claims. That is, indicate support for newly added claim language by specifically pointing to page(s) and line number(s) in the specification and/or drawing figure(s). This will assist the examiner in prosecuting the application.

When responding to this Office Action, applicant is advised to clearly point out the patentable novelty which he or she thinks the claims present, in view of the state of the art disclosed by the references cited or the objections made. He or she must also show how the amendments avoid such references or objections See 37 CFR 1.111(c).

Contact Information

12. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Bai D. Vu whose telephone number is 571-270-1751. The examiner can normally be reached on Mon - Fri 7:30 - 5:00 EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Christian Chace can be reached on 571-272-4190. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Bai D. Vu/
Examiner, Art Unit 2165
12/18/2008

/C. T. T./

Primary Examiner, Art Unit 2169

/Christian P. Chace/

Supervisory Patent Examiner, Art Unit 2165